curl user survey 2021 analysis

“curl is amazing and I've been using it for over 20 years. That's the longest time I have used anything except the fancy drill I got when I was 18” (anonymous)

summary and analysis by Daniel Stenberg

version 1.0 - July 5, 2021

About curl

Curl is a mature and well established open source project that produces the curl tool and the libcurl library. We are a small project, with few maintainers, with little commercial backing and yet we're over 23 years old and we have gathered help from over 2,400 contributors through the years. Our products run in several billion Internet connected devices, applications, tools and services. *curl is one of the world's most widely used software components. Possibly even **the** most widely used component!*

See https://curl.se for everything not answered in this summary.

Survey Background

We do this user survey annually in an attempt to catch trends, views and longer running changes in the project, its users and in how curl fits into the wider ecosystem.

We only reach and get responses from a small subset of users who voluntarily decide to fill in the questionnaire while the vast majority of users and curl developers never get to hear about it and never get an opportunity to respond. Self-selected respondents to a survey makes the results hard to interpret and judge. This should make us ask ourselves: is this what our users think, or is it just the opinions of the subset of users that we happened to reach. We simply have to work with what we have.

This year, the survey was up 14 days from May 24 to and including June 6th. This was the 8th annual survey as the first one ran in 2014.

The survey was announced on the curl-users and curl-library mailing lists (with one reminder), numerous times on Daniel's twitter feed (@bagder) and on Daniel's blog (https://daniel.haxx.se/blog). The survey was also announced on the curl web site with an "alert style" banner on most pages on the site that made it hard to miss for web visitors.

It took a seriously long time for me this time to collect the answers and produce this analysis document. I think I need to change something to make it less of a hassle and time consuming thing in the future.

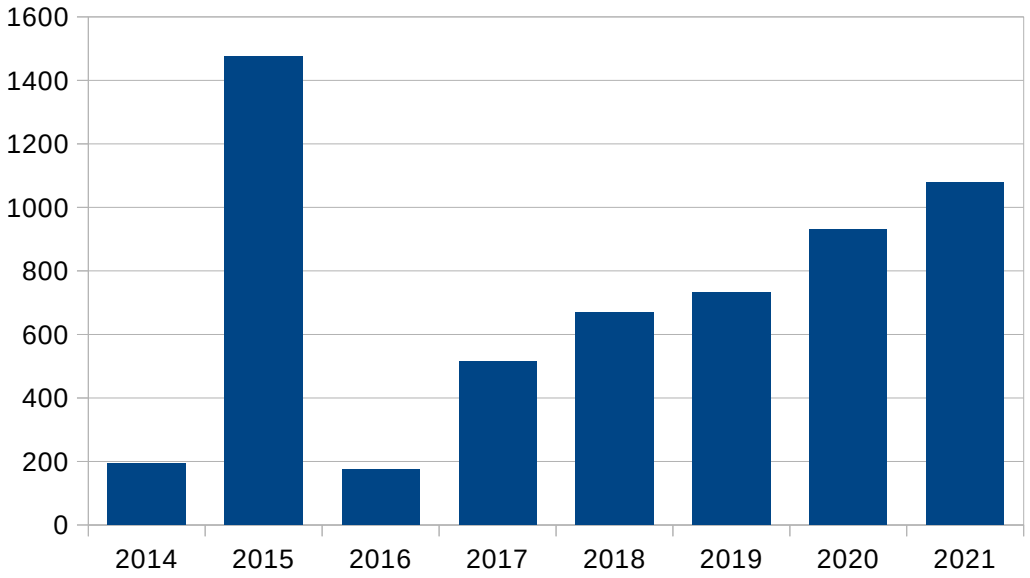
Survey hosted by Google

We use a service run by Google to perform the survey, which leads to us loosing the share of users who refuse to use services hosted by them. We feel compelled to go with simplicity, no cost and convenience of the service rather than trying to please everyone. We have not found a compelling and competitive alternative provider for the survey.

Responses

In order to capture as many opinions and views as possible we're trying every year to reach out as widely as possible and of course we want to get more responses than we did last year. It's not about the quantity really, but reaching out to those users who really have thoughts and ideas about curl.

In 2021 we managed to yet again gather more responses than the previous year. 1,078 persons graciously spent some of their time helping us out. This is up 15% from last year's 930 responses.



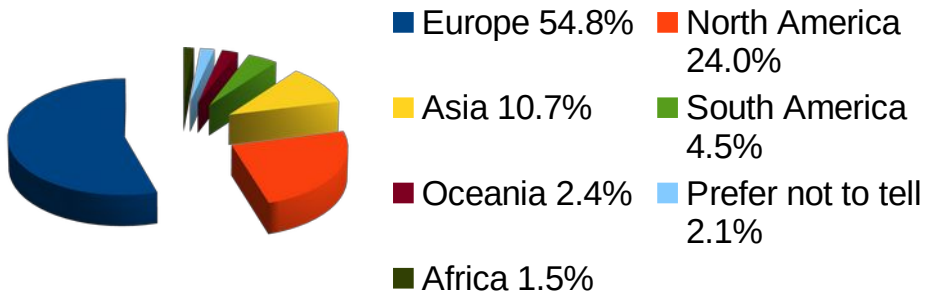
Returning

Only 128 respondents (12%) said they filled out the survey last year. Almost as many, 120, couldn't remember and 75.5% said they didn't.

This is almost the exact proportion of answers as we also got last year. This response rate is interesting primarily when we look at responses that seem to look very similar to previous years as then that distribution is not based on it being the exact same persons responding.

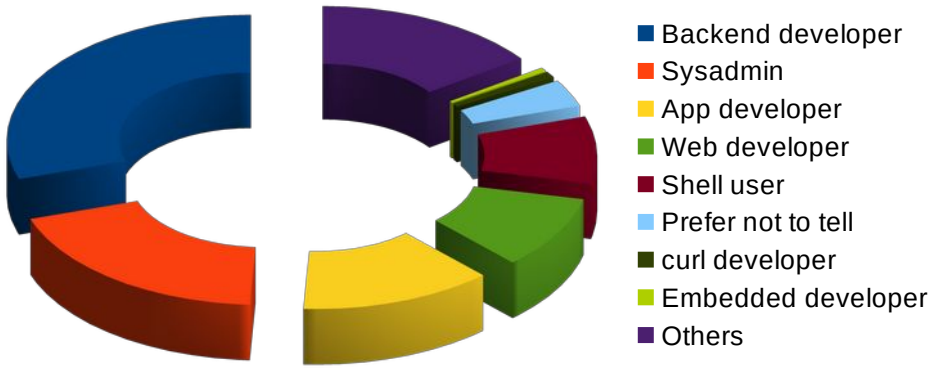
Continents

1051 respondents filled in what continent they're from. Like previous years, Europe is over half and the US about a quarter of the users. I was glad to see that the portion of users from outside Europe and the US now reached over 20% for the first time in a survey, ending at 21.2%.



Kind of users?

This distribution is again also very similar to previous years. 30% are backend developers, 19% are sysadmins and then it trickle down. 13% are app developers and then all the rest are below 10%.



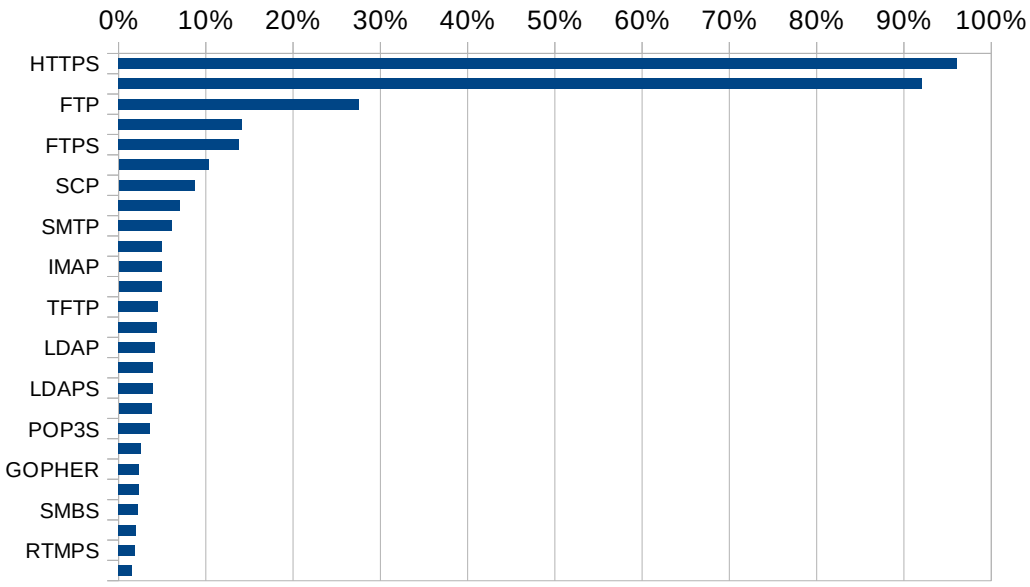
Protocols

(n = 1057)

What protocols are people using? The simple answer is that everyone uses the same protocols to the same extent they've used in previous years as well. HTTPS and HTTP are the protocols curl is known to speak and they're used *far* more than than any other at 96% and 92% respective.

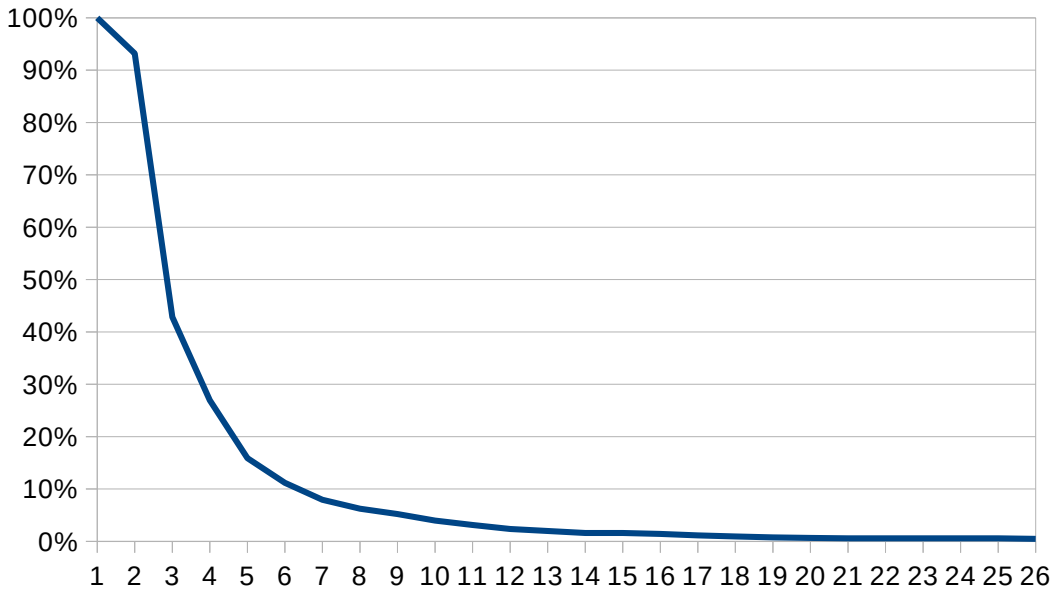
The total amount of supported transfer protocols is officially 26. There are yet again six protocols that are used by 10% or more of the users. Apart from the already mentioned protocols, position 3 to 6 are held by: FTP (27.5%), SFTP (14.2%), FTPS (13.8%) and FILE (10.3%). These top protocols are also ranked in the exact same individual order as last year.

The most recently added protocol (GOPHERS) turned out to be the least used. 16 users said they used it.



On average, 3.4 protocols were used per person. 6 users claimed they had used all 26 protocols.

The median user uses but two protocols but 43% uses three protocols and more than 10% of users use 6!



Platforms

(n= 1062)

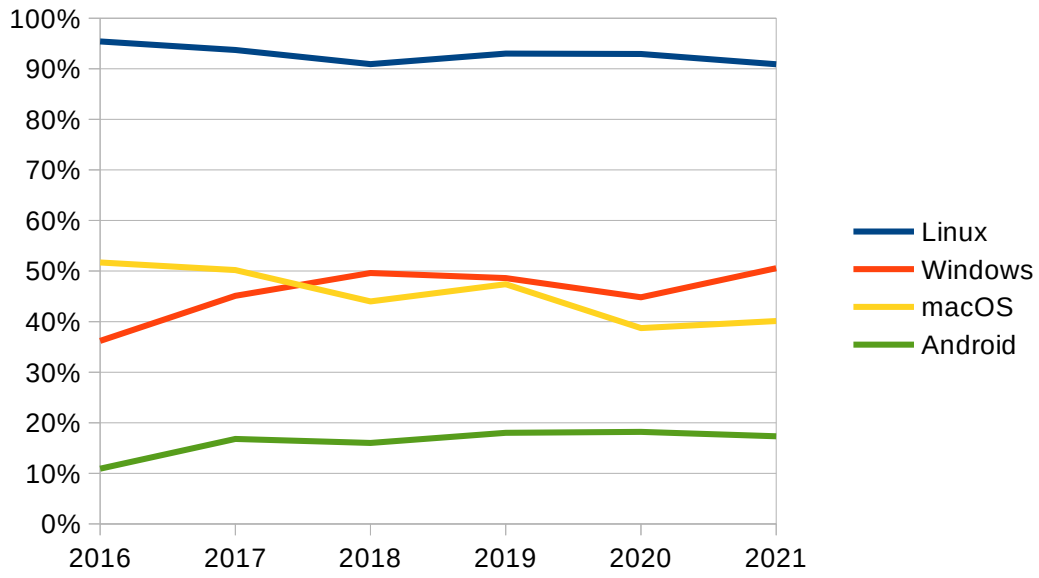
The distribution among platforms remain roughly the same. This is a multi-choice question so lots of users selected multiple options here. More than 9 out of 10 users still used curl on Linux.

Platform	Share 2021
Linux	90.87%

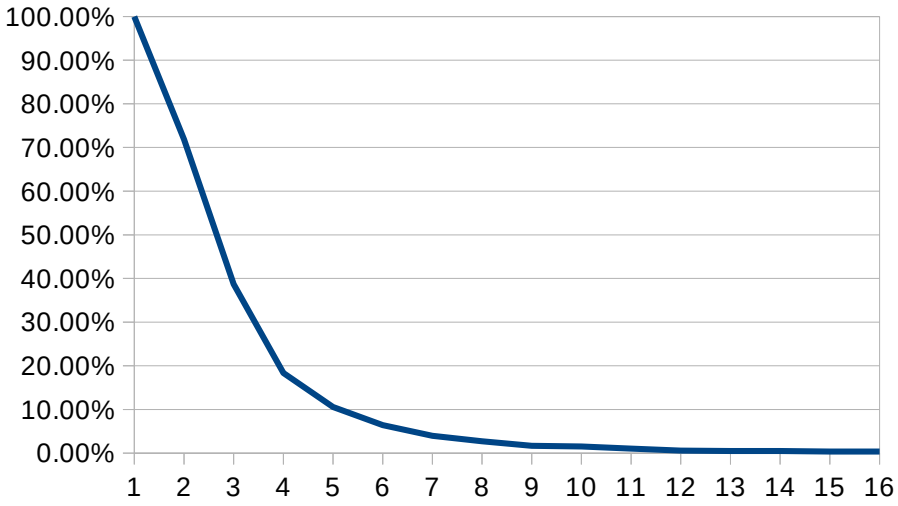
Windows	50.57%
macOS	40.11%
Android	17.33%
FreeBSD	9.23%
iOS	7.72%
OpenBSD	4.61%
Solaris	3.11%
Game console	2.64%
NetBSD	1.98%
Another unix	1.69%
OpenIndiana	1.41%
MS-DOS	1.22%
AIX	1.13%
RTOS	1.04%
VMS	0.85%
IBM i	0.75%
AmigaOS	0.75%
HPUX	0.56%
IRIX	0.56%

There's possibly a trend looking over multiple years that Windows has grown as a curl user platform and macOS has shrunken a little. For the first time Windows reached over 50% (up from 44.8% last year). Both FreeBSD and OpenBSD lost over 4% percentage points. Combined with iOS climbing to its highest level yet, 7.7%, iOS is now a bigger curl platform than OpenBSD.

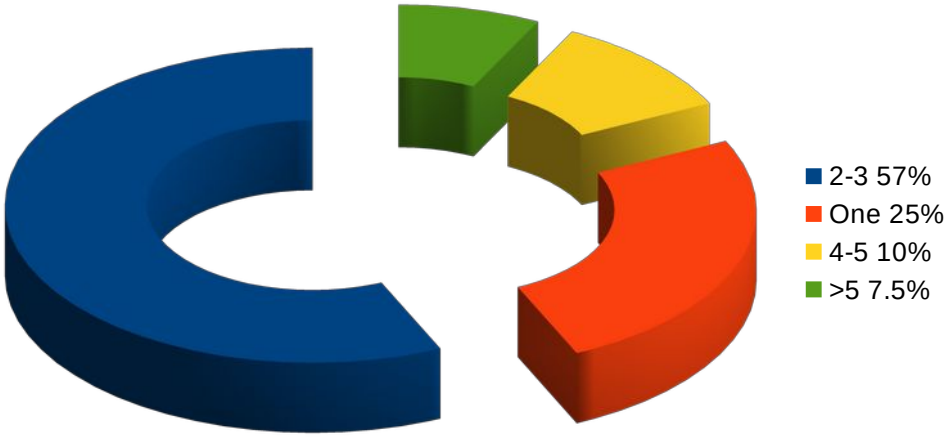
The top-4 curl users platforms over the last 6 years.



According to this question, the number of used platforms among users is distributed like this



That distribution is interesting to compare with how people answer how many platforms then use. 25% specified they use one platform, while the previous question says 80% uses two or more...



This too, closely matches previous years' distributions.

Windows versions

(n = 557)

It's useful to keep track of what Windows versions users are using curl with. This guides the development team on how to do with deprecation of support for old versions etc.

Windows 10 was the only version that grew its share, up from 92.0% last year, which I can only interpret as something good. The really ancient Windows versions are now very small usage wise.

The complete summary of this year's distribution:

Version	**2021**
Windows 10	95.0%
Windows 7	18.5%

Windows Server 2012 / 2016	13.3%
Windows 8	9.3%
Windows XP	5.2%
Windows Server 2008	5.0%
Windows Vista	2.9%
Windows Server 2003	2.2%
Windows 2000	1.1%
Windows CE/Embedded	0.7%
Windows Server 2019	0.7%
Windows 98	0.4%
Windows 95	0.4%

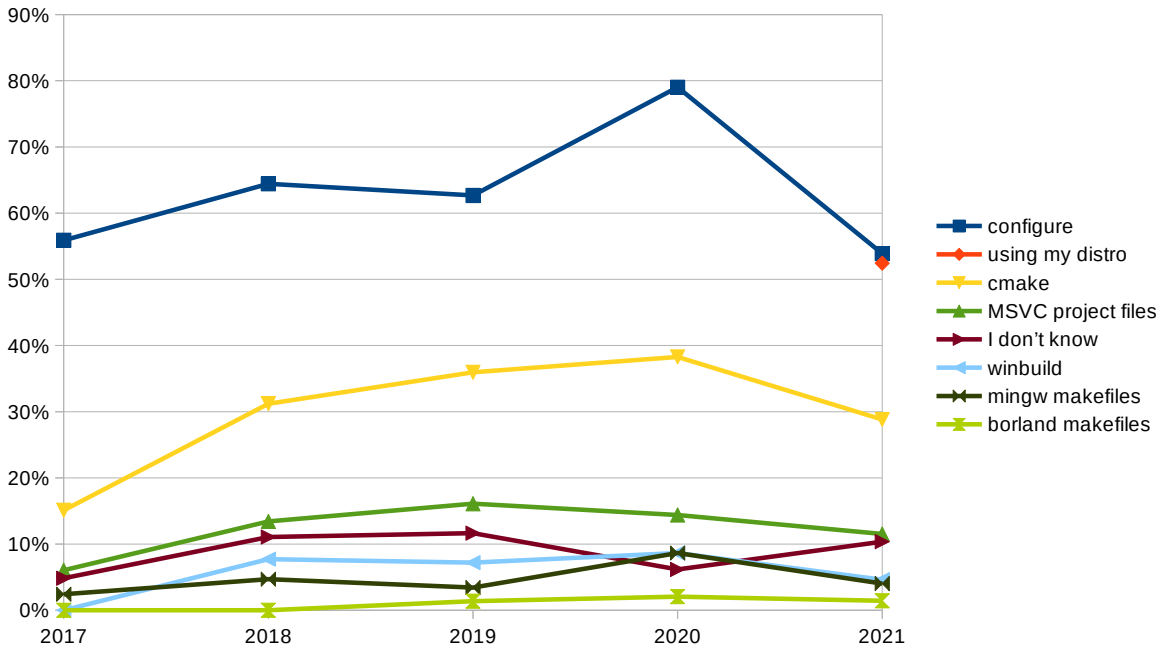
Building curl

(n = 1049)

The idea behind this question is of course to get an idea which of the build methods that are used or not used. New answer option for this year was “yes using my distro” because in previous years a lot of people wrote that in the text field as people in general think of installing packages with distros that build the packages as building curl. I’m not going to debate that, but instead just conclude that a lot of people selected that option.

66.9% of the respondents answered a plain no, down from 75% last year. Meaning a third of the audience build curl.

Configure remains the clear leader even after a huge drop this year and cmake’s growth over the last few years ended. The MSVC project files at 11.5% is the only other option above 10%.



(This graph above is the distribution of the alternatives when “no” has been deducted.)

Features

(n = 891)

Internet trends come and go, and so does what features are popular and used. This question tries to spot such trends among our use base.

What do we learn here this year? We added two features: HTTPS proxy and HSTS and both of them are well-used features. It also turns out that HTTP/2 and HTTP/3 usage grew. **The quickest growing features right now are HTTP/3, DoH and unix domain sockets.**

In the table below I’ve highlighted the features that had a 15% or more increase or shrunk with more than 15%.

Feature	% share of respondents	Change since 2020
HTTP/2	62.7	3.90%
TLS client certificates	34.3	-5.65%
HTTP proxy	30.4	-8.66%
HTTP automatic decompression	27.9	-1.60%
TCP keepalive	19.6	-16.06%
HTTP/3	18.9	32.78%
HTTPS proxy	17.8	N/A
the internal libcurl progress meter	17.8	-12.09%
using libcurl multi-threaded	17.8	-10.77%
SOCKS proxy	17.6	-20.63%
UNIX domain sockets	13.1	17.24%
Bandwidth rate limiting	11.1	-21.20%
HSTS	11.1	N/A
DNS-over-HTTPS (DoH)	11.1	18.20%
.netrc	9.8	-3.32%
curl_multi_socket API	9.2	-17.09%
NTLM auth	8.9	-10.44%
CURLOPT_FAILONERROR	6.8	-8.72%
HTTP/0.9	5.1	-2.87%
Alt-svc	3.4	8.61%
the share interface	2.9	-27.05%
Metalink	1.2	-43.88%

This Metalink feature is being removed in 2021 due to security concerns. The very low usage helped us make that decision.

TLS backends

(n = 1004)

We reached a new record amount of people saying “I don’t know”, at 27% of all answers. To me this is a rather good sign. It means the particular flavor has not been important enough to make a dent in that person’s mind, which could mean that curl’s TLS abstraction layer has worked there and there hasn’t been any significant difference noticed that could be deduced to the TLS flavour.

This year I added two options to the answers: AmiSSL and rustls. The former one was wrongly not included before and the latter is a new backend we support. How do these fare? Let’s take a look at user share with the “I don’t knows” removed.

OpenSSL remains the undisputed leader. Among the most used options, GnuTLS fell significantly while both Secure Transport and Schannel picked up a lot of users. Secure Transport so much that it is now the third most used TLS backend!

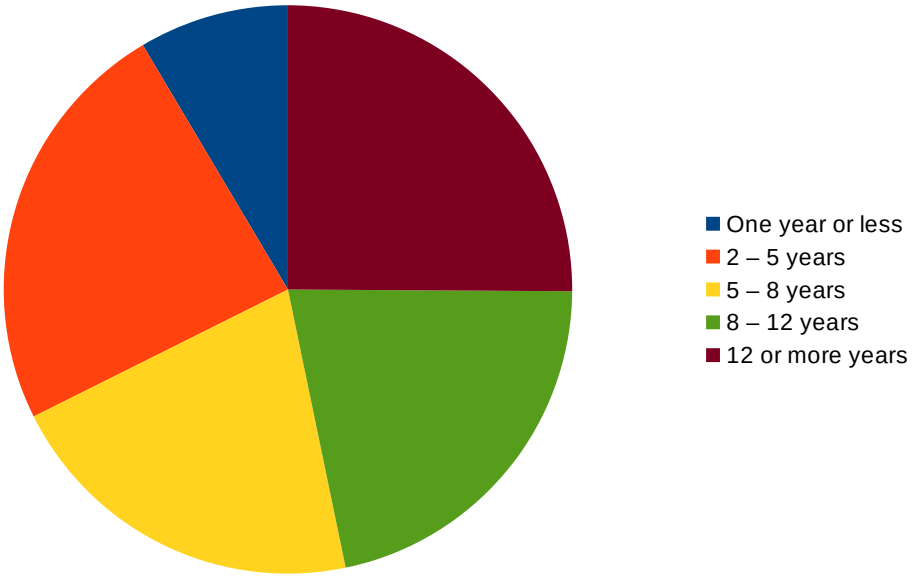
Among the lesser used ones, wolfSSL and gskit bumped their shares a lot. NSS took the biggest hit this year as fewer and fewer curl users are using it.

TLS	**2021**	**2020**	Change year-to-year
OpenSSL	97.4	98.7	-1.3%
GnuTLS	16.0	20.3	-21.1%
Secure Transport	15.6	12.7	23.2%
schannel	14.3	11.7	21.9%
libressl	12.7	14.4	-11.8%
BoringSSL	5.0	4.7	7.1%
mbedTLS	4.4	3.7	17.9%
NSS	4.1	5.6	-26.8%
wolfSSL	2.6	1.6	62.5%
rustls	2		0.0%
BearSSL	1.5	0.0	0.0%
AmiSSL	1.5		0.0%
gskit	0.8	0.3	200.0%
Mesalink	0.4	0.4	0.0%

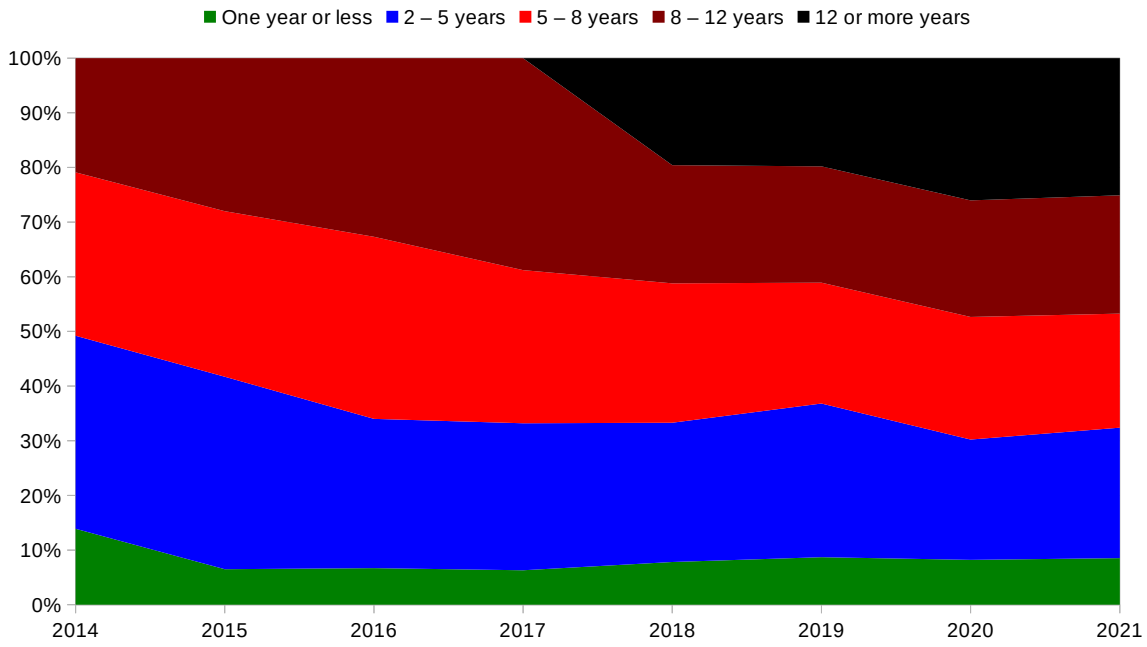
Years of curl use

(n = 1044)

As the project grows older, it isn't unexpected that we get more and more users who have used it for a very long time. Some of you used it already in the 1990s and can thus boast over twenty years of curl. If the share of newcomers would start to shrink significantly we can probably detect that the younglings are abandoning it – but that doesn't seem to happen. The share of reasonably fresh users seem to remain at a stable level in 2021 as it has for the last few years. 25.1% say 12 years or more, 8.5% say one year or less.



The distribution of the respondents over the last years look like this:

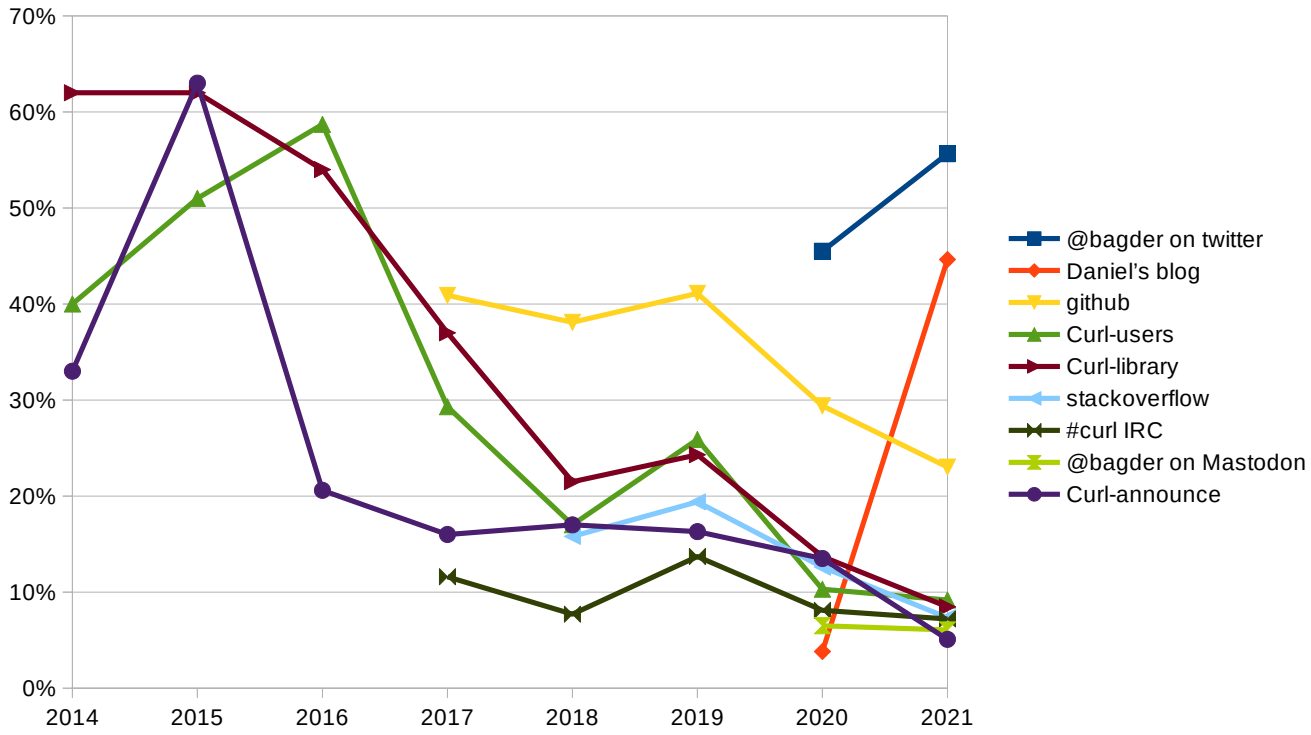


Participating channels

(n = 708)

Where do people learn about new curl things and news and information around the project?

This is a question with an interesting change in answers over time so I'm including a graph below showing all eight years we've asked users. **Daniel's Twitter account and personal blog have become the dominant channels for curl related info.** The blog is at 44.6% up from last years 3.8%!



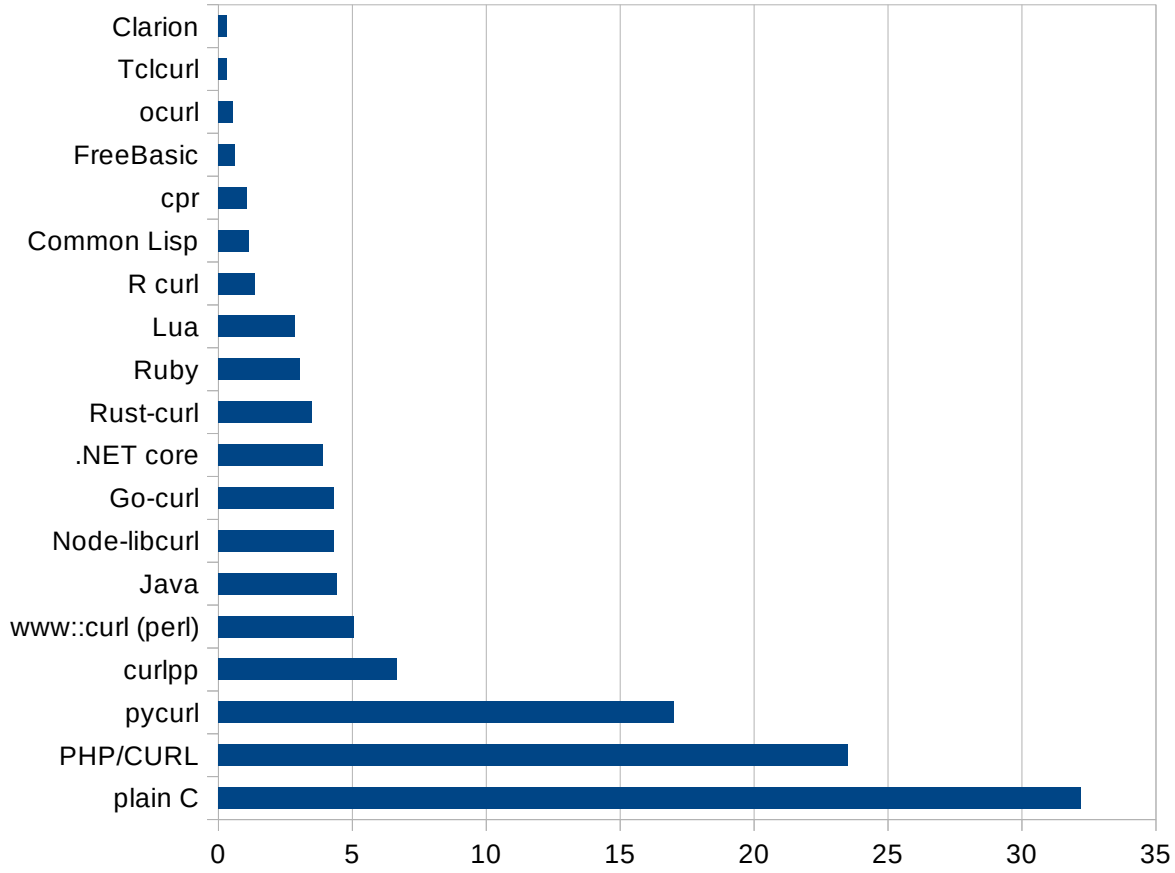
The distribution for 2021

Channel	2021
@bagder on twitter	55.7%
Daniel's blog	44.6%
Curl repository on github	23.0%
Curl-users mailing list	9.2%
Curl-library mailing list	8.5%
stackoverflow	7.3%
#curl IRC	7.2%
@bagder on Mastodon	6.1%
Curl-announce mailing list	5.1%

How do you “access” libcurl

(n = 947)

The distribution remains roughly the same. 75% marked the curl command line tool, but among the different ways to access the libcurl API, they were distributed like this, roughly keeping up the stable trend we’ve seen over the last few years.



Contributions

(n = 904)

What’s the ways people contribute to the project? This question of course also serve as a reminder to people that we’re open source and we rely on help from everyone.

This year doesn’t look much different than previous years, even if I now added an option for “I have curl stickers on prominent places”…

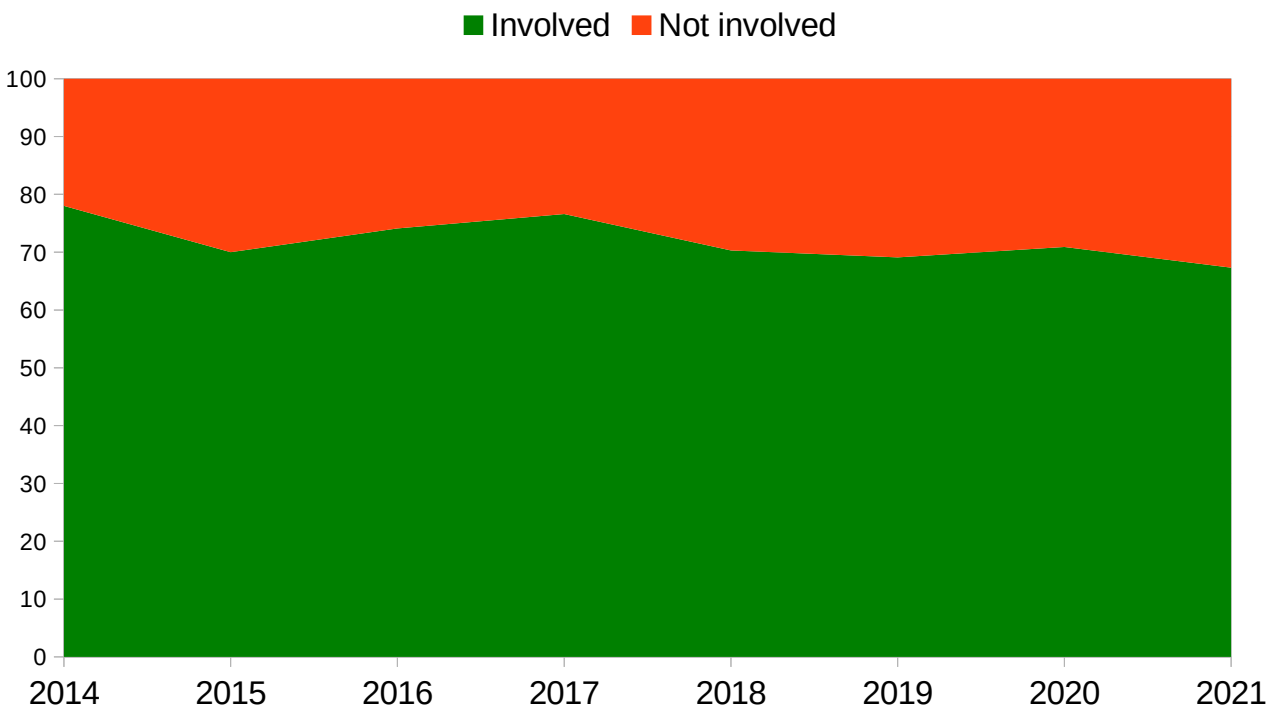
Help	**2021 share**
I haven’t contributed yet	74.2%
I’ve filed bug reports	9.5%
I’ve sent patches or pull requests	8.0%
I can’t remember	5.3%
I’ve helped out in other ways	4.9%
curl stickers on prominent places	4.9%

I’ve responded to mailing lists / forums	4.6%
I’ve donated money	4.0%
I spend time in the IRC channel	2.2%
I run tests or provide infrastructure	1.2%
I wite documentation	1.2%

Other projects

(n = 1035)

There's a slow trend of people answering to this survey being less involved in other projects, but there's still more than two thirds (67.3%) this year involved in other open source projects.



Reasons not to contribute to the project

(n = 994)

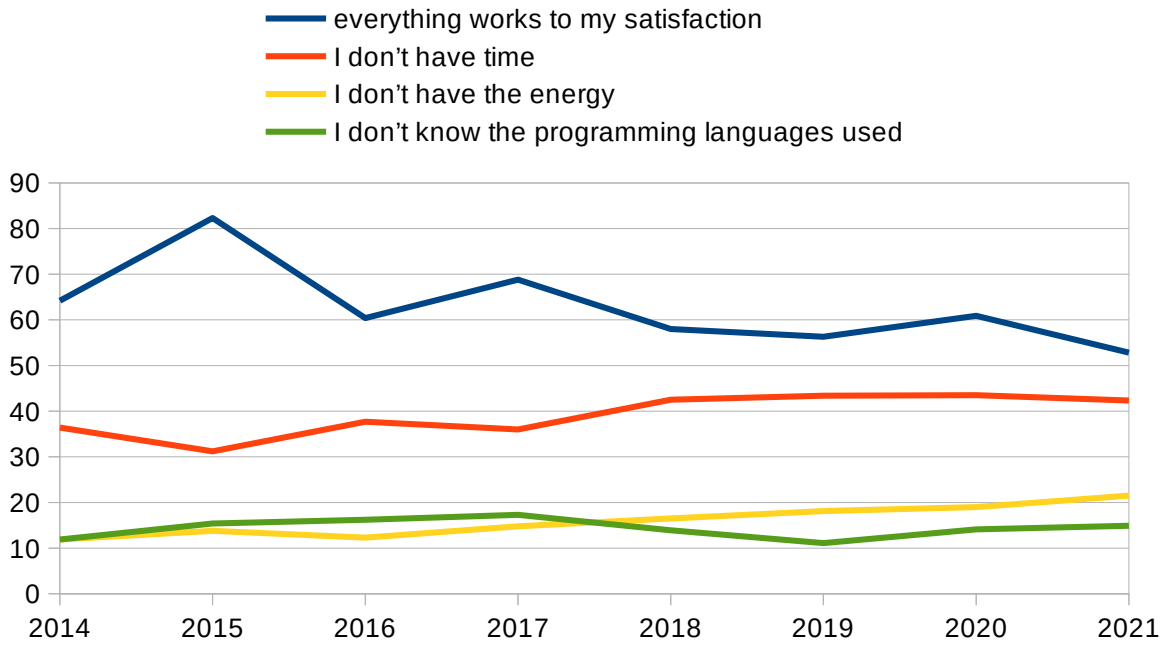
The "everything works to my satisfaction" remains the top "excuse" why people are not contributing to the curl project, down to the all-time low 52.8%. The "I don't have time" remains at a high 42.4% level and the #3 and #4 reasons are also growing this year. Here's the full table

Why not contribute?	Share 2021	Change year-to-year
everything works to my satisfaction	52.8	-13.3%
I don't have time	42.4	-2.6%
I don't have the energy	21.5	13.2%
I don't know the programming languages used	14.9	5.6%
Too hard to get started and figure out where to tweak	14.5	-18.6%
things get fixed fast enough	14.3	-6.6%
my work/legal reasons prohibit me	4.6	-1.5%
I don't like or approve of github	2.7	-38.2%
I can't deal with the tools	2.7	70.0%
I don't like or use email	2.0	18.2%

I find it hard to work with the curl developers	1.5	88.8%
the project doesn’t want my changes	1.1	-14.6%

There are some very large deltas since 2021 if we look at the bottom of the table, but then every answer there gets very few check-marks which means one or two extra makes a very large delta.

Here’s the top-4 explanation development shown over time:



What could the curl project do/change to get (more) contributions from you?

(n = 206)

This was a free-text field which makes it hard both to analyze and to present. About one out of five respondents wrote something.

The answers can generally be divided into these eight different categories:

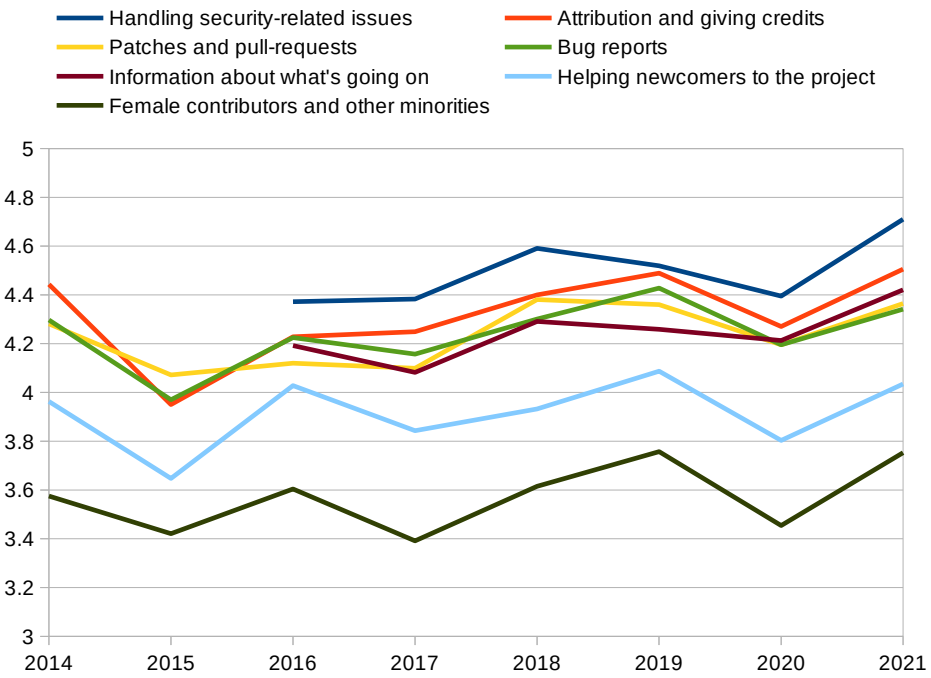
1. I can’t think of anything to add to the project
2. I don’t have the skills to contribute (the code is too hard/big/wrong language)
3. “add bugs for me to fix”
4. “invent a time machine”
5. Mark issues as good starting-points
6. I didn’t know curl need help
7. Use another language
8. Random other stuff that isn’t related to contributions

Category 5: “mark issues as good starting-points” is interesting. I often read how this is suggested as a way to help newcomers into projects. I just don’t know how to do it. How do we determine what’s easy? How do we prevent experienced users from just fixing them immediately (because they are easy) ?

Category 6: “I didn’t know curl need help” is similarly tricky. Several answers urged us to ask for help when we need it and that we should better point out where help is required. Unfortunately they mostly seem to miss the point that in a project like curl we *always* need help and assistance since there are always a range of bugs to fix and quirks to remove. We also provide (huge) lists with known bugs and TODO items. What else can we do?

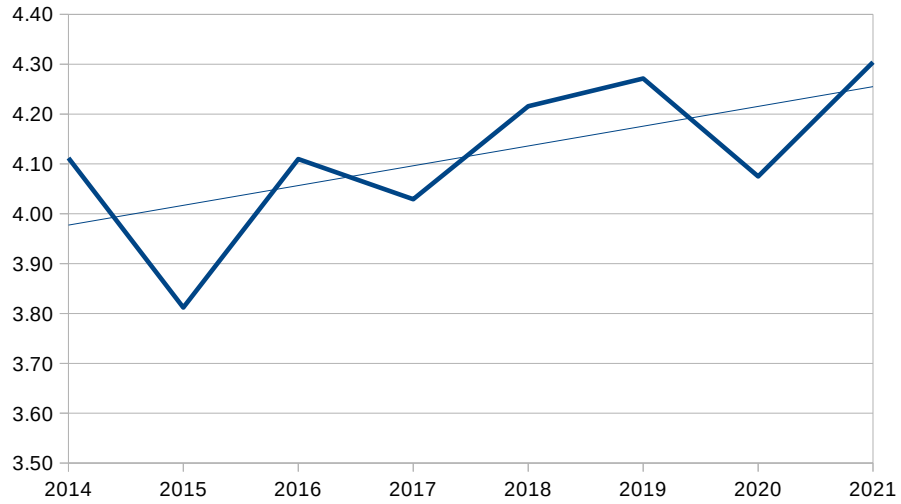
How good is the project to handle…

Asked to grade the project in these seven different areas between 1 to 5, this is designed to track if the project trends in a different direction or perhaps improves or degrades over time. Sgain we get an almost amazingly stable result over time, increasing ever so slightly over time.



A pessimist's view of this would be that all our qualities are in the same relative order after all this time so even if we might've improved slightly over time, there's no big change. And the ones that were our lowest ranked qualities in 2014 are still our lowest ranked qualities…

The general upward trend is best visible when the average of all scores are drawn as a single plot with a corresponding trend line. This year's average of 4.3 is the highest one measured so far.



Which are the curl project's best and worst areas?

(n = 877 for best, 226 for worst)

These two questions list 18 different "areas" in the project and asks the user to select the 3 best and the 3 worst areas. More than three times as many could select best areas as would select worst areas.

The available options to select from were:

- The quality of the products, curl/libcurl
- Its availability and functionality on many platforms
- The support of many protocols
- Documentation
- The libcurl API
- Standards compliance
- The features of the protocol implementations
- Support of multiple SSL backends
- Security
- Footprint of the library/executable
- Project leadership
- Transfer speeds
- The user and developer community
- Bug fix rate
- Project web site and infrastructure
- Welcoming to new users and contributors
- Test suite
- Its build environment/setup

The top-5 voted best areas in the project were the same as recent years with a similar distribution:

the quality of the products, curl/libcurl	57.0%
its availability and functionality on many platforms	55.6%
the support of many protocols	39.8%
documentation	36.6%
the libcurl API	32.0%

When asked to pick the "worst" areas, people also pick the ones they've picked recent years:

documentation	28.3%
its build environment/setup	18.1%
the libcurl API	19.0%
project web site and infrastructure	11.5%
welcoming to new users and contributors	13.3%

It is of course interesting to see "documentation" and the API as usual being featured in both top-lists.

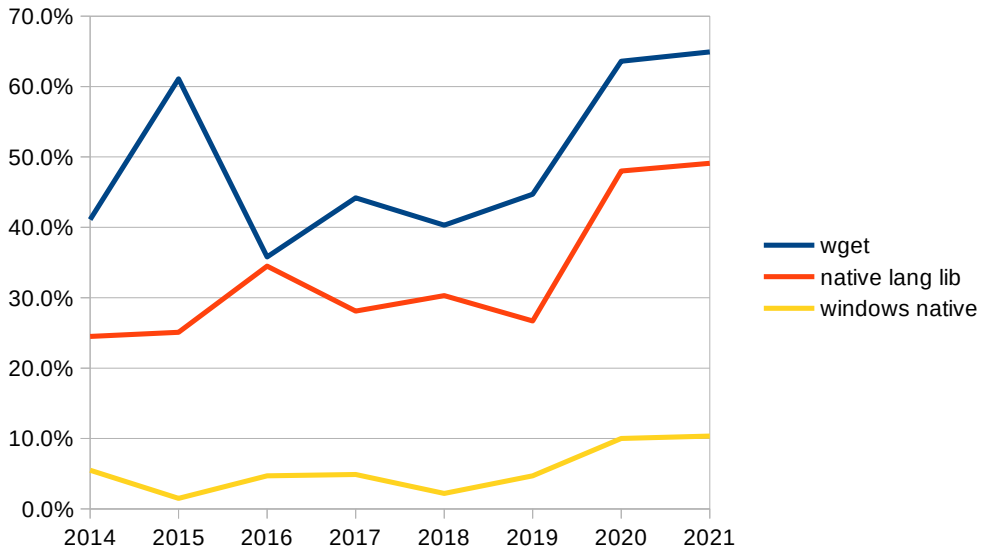
If you couldn't use libcurl, what would be your preferred transfer library alternatives?

(n = 821)

Time for a check on the competition. What would people use instead of libcurl?

When asking the question, I'm curious to find out which libraries that could be considered the main competitors. I'm not convinced everyone who answers the question think of it that way since "wget or code from wget" keeps being the top response year after year and yet I don't think most users actually opt for that when they go with something else than libcurl…

The second-most answer seems more likely then: "a native lib in Perl, Python, Java, Go, Rust, etc" which also climbs somewhat over the years. The 3rd most popular answer "windows native" climbs up to 10% of the respondents.



I think next year we should probably split up the "native lib in.." option into separate ones for the different languages.

Which other download utilities do you normally use?

(n = 957)

No surprises here. The top alternatives are wget, scp/sftp and rsync, used by more than half of curl users. (But note that wget2 is a separate option and that shows it certainly has not "taken off" out there). The alternatives that received more than 5% of the answers were:

wget	74.8%
scp/sftp	61.5%
rsync	56.3%
FileZilla	24.7%

nc	24.0%
ftp	21.4%
winscp	18.8%
httpie	11.3%
fetch	11.3%
Powershell	10.4%
lynx	10.3%
aria2c	10.1%
lftp	7.3%
wget2	7.3%
w3m	5.0%

If you miss support for something, tell us what!

(n = 797)

There's something to be said about asking people what they want in a casual survey. It's very easy to just click a bunch of buttons for everything that sounds even just remotely cool and be done with it.

This is a question for which we've of course subsequently have removed answers from over the year that we've implemented. We also try to add new options for things that have popped up and are being discussed in the community.

This year I added "JSON support" and it *immediately* become the top answer to this question as almost half the population thought this was a good idea. Now, exactly how such support would look like was not detailed and in recent discussions when I've tried to poll the community for what things we could do to curl to improve JSON support (without stepping on jq's toes) I've mostly been told that curl doesn't need more JSON adaptions and that there already are surrounding helper tools that do the JSON part very well. Mixed messages…

JSON	43.54%
websockets	41.03%
DNS-over-TLS	22.58%
rsync	21.83%
gRPC	20.83%
bittorrent	19.70%
GraphQL	16.19%
DNSSEC (DANE)	14.93%
SRV records	12.42%
SMB v2/v3	11.67%
Auto-detect proxy	11.17%
Gemini	10.29%
OCSP	9.16%
DNS: URLs	8.78%
ECH (ex ESNI)	7.90%
AIA (download certs)	7.65%
CoAP	3.01%

The free-text form that allows users to fill in their missing features is a gold-mine of ideas and various wishes – without about a quarter of them asking for something we already support... Some of my favorites this year:

- Per-multi (or per-share) bandwidth limiting
- test-suite running in parallel

- A callback-style socket abstraction for the underlain connection. The way lib curl handle sockets currently is a no-go for windows IOCP based thread pools and etc.
- integration with gpg-agent to allow for encrypted .netrc
- CURLOPT_WILDCARDMATCH should support SFTP
- support for automatic duplicate file renaming.
- Supporting with C++ STL without any dependencies by Boost, Qt or other non-stl libs.
- add a --follow-redirects in curl as an alias for –location
- A standardized C++ API that utilizes the added options for abstractions
- Multi-threaded sftp downloads like lftp
- Browser emulation so a web server can’t distinguish curl from a normal browser
- a 'raw' flat/minimal c build

What feature/bug would you like to see the project remove?

(n = 47)

Maybe not a very useful question. This typically triggers a few users to mention protocols they don’t use and “bloat”. I think I can remove this question next year without it making anything worse.

Which of these API(s) would you use if they existed?

(n = 608)

Again a question that is easy to just pick answers from like a menu that we should interpret casually and not too strict. The read/write API experiment has been provided in the fcurl repository for many years now without getting much user traction and yet 37% this year says that they “would use it”. The JSON option was new this year and a whopping 55.8% said they’d use a JSON API if libcurl provided one...

JSON generation/parsing	55.8%
header parsing/extracting	43.6%
establishing a websocket connection	37.3%
a read()/write() style API for downloading and uploading	28.6%
Server-side support library for HTTP(S)	22.0%
HTTP Content-Disposition header parser/helper for applications	15.6%
Pluggable async DNS resolver	11.8%
Per-multi bandwidth limitation settings	8.4%

PAC support	7.1%

Attend curl-up?

(n = 915)

Asked to see how the interest is distributed, and perhaps where the largest pool of “yes” is located. Yes in Europe got 11%, Yes in North America 4.5% and 2.5% said yes to any location. This suggests that we might continue hosting the main curl up event in Europe for a while more.